Reversible Pushdown Transducers

Bruno Guillon, Martin Kutrib, Andreas Malcher, Luca Prigioniero

Reversible Pushdown Transducers*

Bruno Guillon[a], Martin Kutrib[b], Andreas Malcher[b], Luca Prigioniero[b,c]

[a] *LIMOS, Université Clermont Auvergne, Aubière, France*
[b] *Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany*
[c] *Dipartimento di Informatica, Università degli Studi di Milano, via Celoria 18, 20133 Milan, Italy*

Abstract

Deterministic pushdown transducers are studied with respect to their ability to compute reversible transductions, that is, to transform inputs into outputs in a reversible way. This means that the transducers are also backward deterministic and thus are able to uniquely step the computation back and forth. The families of transductions computed are classified with regard to four types of length-preserving transductions as well as to the property of working reversibly. It turns out that accurate to one case separating witness transductions can be provided. For the remaining case it is possible to establish the equivalence of both families by proving that stationary moves can always be removed in length-preserving reversible pushdown transductions.

Keywords: Pushdown transducers, reversible computations, computational capacity, transduction hierarchies.

1. Introduction

Reversible computational models have earned much attention recently, where one incentive for the study of computational devices performing logically reversible computations is probably the question posed by Landauer of whether logical irreversibility is an unavoidable feature of useful computers. Landauer has demonstrated the physical and philosophical importance of this question by showing that whenever a physical computer throws away information about its previous state it must generate a corresponding amount of entropy that results in heat dissipation (see [2] for further details and references). As one of the first models Turing machines have been investigated towards their ability to work reversibly in the work of Lecerf [3] and Bennett [2], where it is shown

*A preliminary version of this work was presented at the *22nd International Conference on Developments in Language Theory (DLT 2018)* and it is published in [1].

Email addresses: bruno.guillon@uca.fr (Bruno Guillon), kutrib@informatik.uni-giessen.de (Martin Kutrib), andreas.malcher@informatik.uni-giessen.de (Andreas Malcher), prigioniero@di.unimi.it (Luca Prigioniero)

that for every Turing machine an equivalent reversible Turing machine can be constructed. At the other end of the Chomsky hierarchy, Angluin introduced reversible computations in deterministic finite automata (DFA) and showed that reversible DFAs are weaker than DFAs in general [4]. Moreover, it is known that the general model and the reversible one coincide if the input head is two-way [5]. An algebraic characterization of languages accepted by reversible multiple-entry DFAs is obtained in [6]. Recent results on reversible regular languages are given in [7, 8, 9], where different aspects concerning the descriptional complexity and the minimality of reversible (one-way) DFAs are studied. For deterministic pushdown automata, it has been shown that the reversible variant is weaker than the general one [10].

Computational models are not only interesting from the viewpoint of accepting some input, but also from the more applied perspective of transforming some input into some output. For example, a parser for a programming language should not only return the information whether or not the input word can be parsed, but also the parse tree in the positive case. Transductions that are computed by different variants of transducers are studied in detail in the book of Berstel [11]. Deterministic and nondeterministic pushdown transducers are investigated in [12], where also characterizations of pushdown transductions as well as applications to the parsing of context-free languages are given. More recently, iterated finite state transducers have been introduced and studied in [13, 14], transducing variants of stack automata have been considered in [15], two-way transducers with auxiliary memory structures have been investigated in [16] with respect to their decidable problems, and the parallel model of cellular automata has been studied in [17] towards its transducing capabilities.

Reversibility in transducing devices has been investigated recently in [18, 19] for deterministic finite state transducers (DFSTs). In the former paper, reversible variants of (two-way) DFSTs are introduced where the reversibility of the transition function depends on the input only. Based on this definition the authors present constructions for the composition and uniformization of two-way DFSTs. In the latter paper, reversible DFSTs are defined whose transition function depends on the input and the output, since reversibility here is meant to preserve information both on the input and the output side. Hence, reversible DFSTs may be considered as reversible Turing machines (see, for example, [20, 2]) with a one-way input tape and a one-way output tape. The main results of [19] concern reversible DFSTs with length-preserving transductions where it is differentiated between different modes that basically differ by the fact whether or not input and output head have to move synchronously.

In this paper, we complement the investigation of reversible models by introducing reversible deterministic pushdown transducers for which we study length-preserving transductions in Mealy mode, strong mode, weak mode, and bounded delay mode. The definition of the model and of the different modes, which basically differ in possible distances between the input and output head, are given in Section 2. In Section 3, we study the computational capacity in detail and we can draw a complete picture. It can be shown by a detailed construction that the Mealy mode and the strong mode coincide. However, this

holds in the reversible case only and is no longer true for possibly irreversible deterministic pushdown transducers. Apart from this equivalence for all remaining inclusions we present suitable witness transductions that show their properness both in the reversible and the general case. Finally, reversible transductions can be separated from general transductions in every of the four modes. Moreover, we can prove incomparability results in all cases where inclusion cannot be obtained.

2. Preliminaries

We denote the non-negative integers $\{0, 1, 2, \ldots\}$ by $\mathbb{N}$. Let Σ^* denote the set of all words over the finite alphabet Σ, and $\Sigma^{\leq k}$ denote its restriction to words of length at most k, for any $k \geq 0$. The *empty word* is denoted by λ. The length of a word w is denoted by $|w|$ and its *reversal* is denoted by w^R. For the number of occurrences of a symbol a in w we use the notation $|w|_a$. *Inclusions* are denoted by $\subseteq$. For convenience, we use S_x to denote $S \cup \{x\}$, where S is a set and x is an element not belonging to S.

Next, we define reversible pushdown transducers that are basically deterministic finite state transducers which are equipped with a pushdown store. We define this model as usual with two tapes, namely, an output tape and an input tape whose inscription is the input word followed by an endmarker. In the forward computation the pushdown transducer decides its operation depending on the current state, the current input symbol, and the symbol on top of the pushdown store. It may perform a right or a stationary move on the input tape, may push onto or pop from the pushdown, and may rewrite the current tape square on the output tape which implies a right move of the output head. The output tape is initially filled with blank symbols.

Formally, we define a *deterministic pushdown transducer* (DPDT) as a system $M = \langle Q, \Sigma, \Gamma, \Delta, \triangleleft, \bot, \sqcup, \delta, q_0, F\rangle$, where Q is the set of *internal states*, Σ is the set of *input symbols* not containing the *endmarker* $\triangleleft$, Γ is the set of *pushdown symbols* not containing the *bottom-of-pushdown symbol* $\bot$, Δ is the set of *output symbols* not containing the *blank symbol* $\sqcup$, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states*, and

$$\delta \colon Q \times \Sigma_{\triangleleft} \times \Gamma_{\bot} \to Q \times (\{\texttt{pop}, \texttt{top}\} \cup \{\, \texttt{push}(x) \mid x \in \Gamma \,\}) \times \Delta_{\sqcup} \times \{0, 1\}$$

is the partial *transition function*. An image $(q, \texttt{op}, d, m)$ is indicating that the DPDT has to enter the state q while performing an action `op` on the pushdown store (which can be either a `pop`, a `top` (that is, do nothing) or a `push` action), append the symbol d to the output tape if $d \neq \sqcup$ (otherwise, the output head stays in position and no output symbol is written) and move the input-head forward if $m = 1$ (keep it in current position if $m = 0$). Such an image is determined deterministically from a pre-image (p, a, g) which indicates that the automaton is currently in state p, reading the input symbol a (which may be the endmarker $\triangleleft$) with the symbol g on the top of the pushdown ($g = \bot$ if the pushdown store is empty).

It is understood that the head of the input tape never moves beyond the end-marker (that is, a head move m cannot be equal to 1 if the scanned symbol a is $\triangleleft$) and that, in any configuration, the bottom-of-pushdown symbol appears exactly once at the bottom of the pushdown store (that is, the pushdown action `op` is different from `pop` if the top of the pushdown g is $\perp$).

A *configuration* of a DPDT M is a quintuple (u, p, v, γ, β), where $u \in \Sigma^*$ is the already read part of the input to the left of the input head, $p \in Q$ is the current state, $v \in \Sigma^*\triangleleft$ is the still remaining part of the input, $\gamma \in \Gamma^*\perp$ is the current content of the pushdown store, the leftmost symbol being the top symbol, and $\beta \in \Delta^*$ is the already written part of the output. The *initial configuration* for input w is set to $(\lambda, q_0, w\triangleleft, \perp, \lambda)$. During the course of its computation, M runs through a sequence of configurations. One step from a configuration to its *successor configuration*, denoted by $\vdash$, is defined as follows.

Let $(u, p, av, g\gamma, \beta)$ be a configuration for $p \in Q$, $u \in \Sigma^*$, $av \in \Sigma^*\triangleleft$ with $|a| = 1$, $g\gamma \in \Gamma^*\perp$ with $|g| = 1$, and $\beta \in \Delta^*$. Then we define:

$$
\begin{aligned}
&(u, p, av, g\gamma, \beta) \vdash (ua, q, v, \gamma'\gamma, \beta d) && \text{if } \delta(p, a, g) = (q, \texttt{op}, d, 1);\\
&(u, p, av, g\gamma, \beta) \vdash (ua, q, v, \gamma'\gamma, \beta) && \text{if } \delta(p, a, g) = (q, \texttt{op}, \sqcup, 1);\\
&(u, p, av, g\gamma, \beta) \vdash (u, q, av, \gamma'\gamma, \beta d) && \text{if } \delta(p, a, g) = (q, \texttt{op}, d, 0);\\
&(u, p, av, g\gamma, \beta) \vdash (u, q, av, \gamma'\gamma, \beta) && \text{if } \delta(p, a, g) = (q, \texttt{op}, \sqcup, 0);
\end{aligned}
$$

where $d \in \Delta$ and γ' is equal to λ, g or xg respectively if `op` is equal to `pop`, `top`, or `push`(x) for some $x \in \Gamma$. The reflexive transitive closure of $\vdash$ is denoted by $\vdash^*$. A step is said to be *stationary* if neither the input head is moved nor an output symbol is emitted. A corresponding transition of δ is a *stationary transition*.

A DPDT *halts* if the transition function is undefined for the current configuration. The *output* written by a DPDT M on input $w \in \Sigma^*$ is denoted by $M(w) \in \Delta^*$ and is defined as $M(w) = \beta$, if $(\lambda, q_0, w\triangleleft, \perp, \lambda) \vdash^* (w, q, \triangleleft, \gamma, \beta)$ with $\gamma \in \Gamma^*\perp$, $q \in F$, and M halts. Otherwise, $M(w)$ is undefined. Now, the *transduction* defined by M is the set

$$T(M) = \{\, (w, M(w)) \mid w \in \Sigma^* \text{ and } M(w) \text{ is defined} \,\}.$$

We remark that M may also be considered as a partial function mapping some $w \in \Sigma^*$ to $v \in \Delta^*$. If we build the projection on the first components of $T(M)$, denoted by $L(M)$, then the transducer degenerates to a deterministic language acceptor. In general, the family of all transductions performed by some device of type X is denoted by $\mathscr{T}(X)$.

Now, we turn to *reversible* DPDTs. Basically, reversibility is meant with respect to the possibility of stepping the computation back and forth. So, the machines have also to be backward deterministic. That is, any configuration must have at most one predecessor which, in addition, is computable by a device of the same type. In particular, the machines reread the symbols which they have read in a preceding forward computation step. So, for reverse computation steps the input and output heads are either moved to the *left* or stay stationary.

Figuratively, one can imagine that in a forward step, first the current symbols are read and then the heads are moved, whereas in a backward step first the heads are moved and then the symbols are read.

A DPDT is *reversible* (REV-DPDT) if for any two *distinct* transitions

$$\delta(p, a, g) = (q, \mathtt{op}, d, m) \text{ and } \delta(p', a', g') = (q', \mathtt{op}', d', m')$$

if $q = q'$, then the following conditions are satisfied:

1. $m = m'$ and $(d = \sqcup \iff d' = \sqcup)$;
2. depending on the pushdown operations $\mathtt{op}$ and $\mathtt{op}'$:
   (a) if $\mathtt{op} = \mathtt{pop}$ or $\mathtt{op}' = \mathtt{pop}$, then $(a, d) \neq (a', d')$;
   (b) if $\mathtt{op} = \mathtt{op}' = \mathtt{push}(x)$ for some $x \in \Gamma$, then $(a, d) = (a', d')$ implies $p = p'$;
   (c) otherwise, $(a, d, h) \neq (a', d', h')$ where $h = g$ if $\mathtt{op} = \mathtt{top}$ and $h = x$ if $\mathtt{op} = \mathtt{push}(x)$ for some $x \in \Gamma$ (respectively $h' = g'$ if $\mathtt{op}' = \mathtt{top}$ and $h' = x'$ if $\mathtt{op}' = \mathtt{push}(x')$ for some $x' \in \Gamma$).

Condition 1 means that transitions reaching the same state have to move both heads in the same way. Condition 2 ensures that for any configuration the predecessor configuration can uniquely be determined from the state (which then implies the head movements), the input symbol read and the output symbol written, if at least one transition pops from the pushdown (Condition 2a). The same is ensured if both transitions push the same symbol while the input symbol read and the output symbol written are the same (Condition 2b). In this case, the predecessor states are identical and the only difference between the transitions might be the symbols g and g' on the top of the stack in the pre-image configuration. Finally, the predecessor configuration can uniquely be determined from the state, the input symbol read, the output symbol written, and the symbol on top of the stack (in the image configuration), otherwise (Condition 2c).

A consequence of the definition of reversibility is that *any* configuration of a REV-DPDT has at most one predecessor configuration. In [10] reversible pushdown automata are considered that are required to be reversible only on *reachable* configurations that appear in any computation that starts from an initial configuration. This gives a wider class of automata. However, to justify the notion of reversibility, in [21] it has been shown that it makes a difference for machines, that is, there are machines that are reversible on reachable but not on all configurations, but it does not make a difference for languages which notion is chosen. So, from the perspective of languages and language classes it is safe to stick with either notion of reversibility.

In this paper, we start the investigation of reversible DPDTs and limit ourselves to DPDTs that are *length-preserving*, that is, transducers M such that $|w| = |M(w)|$ for each input w where $M(w)$ is defined. It will turn out that even length-preserving transductions are sufficient for our separations. We differentiate between the following notions. We call a DPDT a *Mealy* pushdown transducer (M-DPDT) if in every step but possibly a last stationary one on the

endmarker an input symbol is read, an output symbol is written, and both heads proceed one position to the right. We call a DPDT *strongly length-preserving* (s-DPDT) if both heads are moved synchronously, that is, either both move rightward or both stay stationary. A DPDT is said to be of *bounded delay* (bd-DPDT) if in any configuration $(u, p, v\triangleleft, \gamma, \beta)$ that evolves from some initial configuration $(\lambda, q_0, uv\triangleleft, \bot, \lambda)$ the difference $\big|\,|u| - |\beta|\,\big|$ is bounded by some fixed constant. Finally, unrestricted (length-preserving) DPDTs are said to be *weakly length-preserving* (w-DPDT).

In order to clarify the definitions we present an example.

Example 1. The injective and length-preserving transduction

$$\tau = \left\{ (a^n \# b^m \# c^\ell \# a^n, a^n \# b^{2\ell} \# c^{m-\ell} \# a^n) \mid m, n \geq 1 \text{ and } 0 \leq \ell < m \right\}$$

can be computed by some REV-w-DPDT

$$M = \langle \{q_0, q_1, q_2, q_3, q_4, q_5, q_f\}, \{a, b, c, \#\}, \{a, b\}, \{a, b, c, \#\}, \triangleleft, \bot, \sqcup, \delta, q_0, \{q_f\} \rangle$$

where the transition function δ is as follows (see Figure 1):

$$\begin{array}{lcl}
\delta(q_0, a, \bot) &=& (q_0, \texttt{push}(a), a, 1), \\
\delta(q_0, a, a) &=& (q_0, \texttt{push}(a), a, 1), \\
\delta(q_0, \#, a) &=& (q_1, \texttt{top}, \sqcup, 1), \\
\delta(q_1, b, a) &=& (q_1, \texttt{push}(b), \sqcup, 1), \\
\delta(q_1, b, b) &=& (q_1, \texttt{push}(b), \sqcup, 1), \\
\delta(q_1, \#, b) &=& (q_2, \texttt{top}, \#, 1), \\
\delta(q_2, c, b) &=& (q_3, \texttt{top}, b, 0), \\
\delta(q_2, \#, b) &=& (q_4, \texttt{top}, \#, 0), \\
\delta(q_3, c, b) &=& (q_2, \texttt{pop}, b, 1), \\
\delta(q_4, \#, b) &=& (q_4, \texttt{pop}, c, 0), \\
\delta(q_4, \#, a) &=& (q_5, \texttt{top}, \#, 1), \\
\delta(q_5, a, a) &=& (q_5, \texttt{pop}, a, 1), \\
\delta(q_5, \triangleleft, \bot) &=& (q_f, \texttt{top}, \sqcup, 0).
\end{array}$$

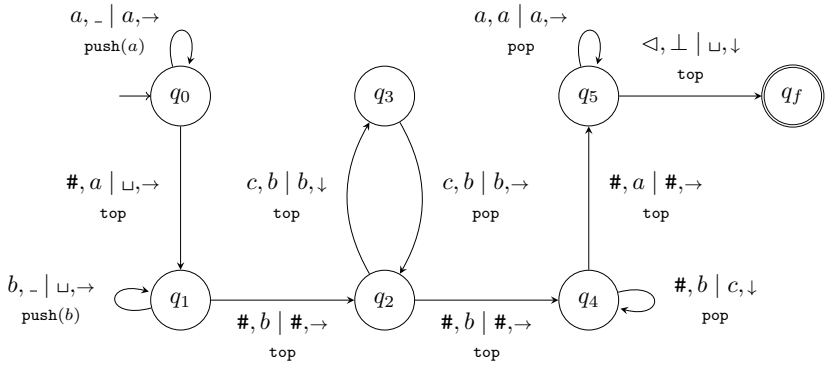

Figure 1: A REV-w-DPDT realizing the transduction τ (Example 1). Edges corresponding to transitions $\delta(p, a, g) = (q, \texttt{op}, d, m)$ are labeled $a, g \mid d, m$, where the symbol _ stands for "any pushdown symbol", $\downarrow$ stands for $m = 0$, and $\rightarrow$ stands for $m = 1$. The pushdown operation op is indicated in the second line of a label (technically speaking, an edge using the symbol _ represents several transitions: one with the empty pushdown symbol $\bot$, the other ones with the symbols from Γ).

The reversibility of M is easily verified by inspecting the transition function and checking the conditions of the definition. In addition, the transduction τ is clearly injective. We note that the projection of τ to the first component is accepted by some reversible pushdown automaton. ■

3. Computational Capacity

We turn to consider the computational capacity of reversible DPDTs. In particular, whenever two types of devices have different language acceptance power, then trivial transductions applied to a language from their symmetric difference would be a witness for separating also the power of the transducers. However, in the following we consider transductions of languages that are accepted by both types of devices in question. In this way, we are in fact separating the capabilities of computing transductions.

3.1. The Role Played by Stationary Transitions

We start with a normalization result stating that stationary moves can be eliminated in every length-preserving reversible DPDT. To this end, the following lemma is used.

Lemma 2 *Every length-preserving reversible DPDT can be converted into an equivalent length-preserving reversible DPDT of the same type that never performs more than m consecutive steps without moving its input head or emitting an output symbol in any halting computation, where $m \geq 0$ is a constant.*

Proof. Let $M = \langle Q, \Sigma, \Gamma, \Delta, \lhd, \bot, \sqcup, \delta, q_0, F\rangle$ be a length-preserving reversible DPDT. We show how M can be converted into an equivalent length-preserving reversible DPDT $M' = \langle Q, \Sigma, \Gamma, \Delta, \lhd, \bot, \sqcup, \delta', q_0, F\rangle$ of the same type having the required property. To this end, the transition function δ is modified with respect to stationary transitions as follows.

(1) Two consecutive `top`-transitions are merged into one. That is, every pair of the form $\delta(p, a, g) = (q, \mathtt{top}, \sqcup, 0)$ and $\delta(q, a, g) = (r, \mathtt{top}, \sqcup, 0)$ is replaced by $\delta'(p, a, g) = (r, \mathtt{top}, \sqcup, 0)$. Since M is deterministic every application of the first transition is followed by an application of the second transition, and since M is reversible every application of the second transition is preceded by an application of the first transition. In fact, the conditions for being reversible are satisfied. This can be seen as follows: Let $\delta'(p', a', g') = (r, \mathtt{op}, d', m')$ be another transition that yields state r. Then $m' = 0$ and $d' = \sqcup$ since M is reversible. Moreover, if $\mathtt{op} = \mathtt{pop}$, then $(a, \sqcup) \neq (a', \sqcup)$ since M is reversible. Otherwise, we have $(a, \sqcup, g) \neq (a', \sqcup, h')$, where h' equals g' or x, respectively if $\mathtt{op} = \mathtt{top}$ or $\mathtt{op} = \mathtt{push}(x)$ for some x. This implies that the transitions $\delta'(p, a, g) = (r, \mathtt{top}, \sqcup, 0)$ and $\delta'(p', a', g') = (r, \mathtt{op}, d', m')$ satisfy the condition for reversibility. So, the construction step preserves reversibility and yields an equivalent automaton.

(2) A `push`-transition and a following `top`-transition are merged into one `push`-transition. That is, every pair of the form $\delta(p, a, g) = (q, \mathtt{push}(x), \sqcup, 0)$

and $\delta(q,a,x) = (r, \texttt{top}, \sqcup, 0)$ is replaced by $\delta'(p,a,g) = (r, \texttt{push}(x), \sqcup, 0)$. As before, the construction step clearly yields an equivalent automaton and preserves reversibility. Namely, let $\delta'(p',a',g') = (r, \texttt{op}, d', m')$ be another transition that yields state r. Then $m' = 0$ and $d' = \sqcup$ since M is reversible. Moreover, if $\texttt{op} = \texttt{push}(x)$ then $(a, \sqcup, x) \neq (a', \sqcup, x)$ since M is reversible. So, $a \neq a'$ and, trivially, $(a, \sqcup) = (a', \sqcup)$ implies $p = p'$. If $\texttt{op} = \texttt{pop}$ then $(a, \sqcup) \neq (a', \sqcup)$ since M is reversible. Thus, $a \neq a'$. If $\texttt{op} = \texttt{top}$ then $(a, \sqcup, x) \neq (a', \sqcup, g')$ since M is reversible. Similarly, if $\texttt{op} = \texttt{push}(y)$ then $(a, \sqcup, x) \neq (a', \sqcup, y)$. So, in any case for $\texttt{op}$ the transitions $\delta'(p,a,g) = (r, \texttt{push}(x), \sqcup, 0)$ and $\delta'(p',a',g') = (r, \texttt{op}, d', m')$ satisfy the condition for reversibility.

(3) A push-transition and a following pop-transition are merged into one top-transition. That is, every pair of the form $\delta(p,a,g) = (q, \texttt{push}(x), \sqcup, 0)$ and $\delta(q,a,x) = (r, \texttt{pop}, \sqcup, 0)$ is replaced by $\delta'(p,a,g) = (r, \texttt{top}, \sqcup, 0)$. To see that the new transition preserves reversibility we assume that there is another transition $\delta'(p',a',g') = (r, \texttt{op}, d', m')$. As before, we conclude $m' = 0$ and $d' = \sqcup$ since M is reversible. Moreover, since $\delta(q,a,x)$ is a pop-transition, we know $(a, \sqcup) \neq (a', \sqcup)$ since M is reversible. Thus, $a \neq a'$, and the transitions $\delta'(p,a,g) = (r, \texttt{top}, \sqcup, 0)$ and $\delta'(p',a',g') = (r, \texttt{op}, d', m')$ satisfy the condition for reversibility.

Next, the three steps are repeated until no more merging is possible, which concludes the construction of M'.

It remains to be shown that there is a constant $m \geq 0$ such that the REV-DPDT M' never performs more than m consecutive steps without moving its input head or emitting an output symbol in any halting computation.

Due to the construction, any sequence of consecutive stationary transitions in any computation of M' possibly starts with a sequence of pop- and top-moves, where no two top-moves appear consecutively. Then several push-moves may follow. After a push-move there is never a pop- or top-move.

Assume that there is a computation on some input such that at the beginning of a sequence of stationary transitions on some input symbol $a \in \Sigma_{\lhd}$ at least $|Q| \cdot |\Gamma| + 1$ consecutive pop- or top-moves are performed. If these steps appear before any non-stationary transition, M' starts each computation with an infinite loop on input $a\Sigma^* \lhd$ or $\lhd$ and, thus, does not halt.

Next, assume that at least $|Q| \cdot |\Gamma| + 1$ consecutive pop- or top-moves appear after some non-stationary transition and define the mapping $\rho : \Sigma^* \times Q \times \Sigma^* \lhd \times \Gamma^* \bot \times \Delta^* \to Q \times \Gamma_\bot$ that maps a configuration to its state and the topmost pushdown symbol. Then there is a (partial) computation $c_{k-1} \vdash c_k \vdash^* c_{k+i} \vdash^* c_{k+i+j-1} \vdash c_{k+i+j}$, where the transition from c_{k-1} to c_k reads some input and moves the input head and all the other transitions are stationary. Moreover, we have $\rho(c_{k+i}) = \rho(c_{k+i+j})$, for some minimal $0 \leq i$, $1 \leq j$ such that $i + j \leq |Q| \cdot |\Gamma| + 1$. Then, for $i = 0$, the transition leading from c_{k-1} to c_k yields the same state as the transition leading from c_{k+j-1} to c_{k+j}. However, the former moves the input head while the latter does not. So, the computation is not reversible.

For $i \geq 1$ we know that $\rho(c_{k+i-1})$ and $\rho(c_{k+i+j-1})$ are different since i has been chosen to be minimal. Let $(q,x) = \rho(c_{k+i}) = \rho(c_{k+i+j})$ and assume

that $\delta'(p,a,g) = (q, \mathtt{op}, \sqcup, 0)$ is the transition leading from c_{k+i-1} to c_{k+i} and $\delta'(p',a,g') = (q, \mathtt{op}', \sqcup, 0)$ is the transition leading from $c_{k+i+j-1}$ to c_{k+i+j}. By assumption, op and op′ are either pop- or top-transitions. If one of them is a pop-transition then $(a, \sqcup)$ has to be different from $(a, \sqcup)$ in order to be reversible, a contradiction. If both transitions are top-transitions $(a, \sqcup, g)$ has to be different from $(a, \sqcup, g')$. However, since both transitions are top-transitions we have $g = x = g'$, a contradiction.

Therefore, any sequence of consecutive stationary transitions starts with at most $|Q| \cdot |\Gamma|$ pop- or top-moves. If there are at least $|Q| \cdot |\Gamma| + 1$ subsequent push-moves, the computation runs into an infinite loop with stationary transitions and, thus, M' does not halt. If, otherwise, there are strictly less than $|Q| \cdot |\Gamma| + 1$ push-moves, the length of the whole sequence of stationary transitions is bounded by the constant $m = 2 \cdot |Q| \cdot |\Gamma|$ that depends on M' only. □

The next goal is to perform all constantly many stationary transitions in a sequence together with a non-stationary transition at once. To this end, the problem that the initial state can be reachable in stationary moves has to be overcome. Otherwise, the reversibility can not be guaranteed since the initial configuration and some configuration that leads to the initial configuration may have a common successor configuration.

Lemma 3 *Every length-preserving reversible DPDT whose initial state is reachable by stationary transitions can be converted into an equivalent length-preserving reversible DPDT of the same type that never performs more than m consecutive stationary moves in any halting computation, where $m \geq 0$ is a constant, and that never enters an initial configuration again.*

Proof. Let $M = \langle Q, \Sigma, \Gamma, \Delta, \lhd, \bot, \sqcup, \delta, q_0, F\rangle$ be a length-preserving reversible DPDT. By Lemma 2 we may assume that M satisfies the first condition. Since M is reversible an inspection of the transition function shows whether q_0 is reachable by stationary transitions. If not, nothing has to be done.

Now assume that q_0 is reachable by stationary transitions. Due to the reversibility, it is reachable only by stationary transitions. The idea of the construction of an equivalent length-preserving reversible DPDT of the same type $M' = \langle Q \cup \{p_0\}, \Sigma, \Gamma \cup \{\triangle\}, \Delta, \lhd, \bot, \sqcup, \delta', p_0, F\rangle$ with the required second property is as follows. A new pushdown symbol $\triangle$ is used that acts similar as the bottom-of-pushdown symbol. Whenever the initial state is entered by a top or pop transition, now a new state p_0 is entered that tests if the pushdown is 'empty'. If so, it pushes a new copy of $\triangle$ and changes to state q_0. In this way, the initial configuration at the beginning of the computation is unique since it is the only one with $\bot$ on the top of the pushdown. Accordingly, the transition function δ is modified as follows.

First, in all transitions the bottom-of-pushdown symbol is replaced by $\triangle$. This preserves the reversibility. Next, it is observed that, for any input symbol a, transitions of the form $\delta(p,a,g) = (q_0, \mathtt{top}, \sqcup, 0)$ and transitions of the form $\delta(p,a,g) = (q_0, \mathtt{pop}, \sqcup, 0)$ cannot exist at the same time. If both types would

exist, the fact $(a, \sqcup) = (a, \sqcup)$ would violate the reversibility. Now the new initial state p_0 is involved by adding the transitions

$$\begin{array}{ll} (1) & \delta'(p_0, a, \bot) = (q_0, \texttt{push}(\triangle), \sqcup, 0), \\ (2) & \delta'(p_0, a, \triangle) = (q_0, \texttt{push}(\triangle), \sqcup, 0), \text{ and} \\ (3) & \delta'(p_0, a, g) = (q_0, \texttt{top}, \sqcup, 0), \end{array}$$

for all $a \in \Sigma$ and $g \in \Gamma$. Transitions of the form $\delta(p, a, \triangle) = (q_0, \texttt{top}, \sqcup, 0)$ are replaced by (4) $\delta'(p, a, \triangle) = (p_0, \texttt{top}, \sqcup, 0)$, and transitions of the form $\delta(p, a, g) = (q_0, \texttt{pop}, \sqcup, 0)$ are replaced by (5) $\delta'(p, a, g) = (p_0, \texttt{pop}, \sqcup, 0)$ (recall that only transitions of one of the forms may exist).

The construction clearly does neither change the transduction computed, nor does it change the type of the transducer. It remains to be shown that the construction preserves reversibility. Transitions (1) and (2) are reversible since the predecessor state is p_0 in both cases. Since $\triangle \neq g$, transitions (1) and (3) are reversible, as well as transitions (2) and (3). Further transitions entering state q_0 must be `push`-transitions that push symbols different from $\bot$ and $\triangle$. So, the reversibility is preserved with respect to these transitions as well. The transitions entering state p_0, namely transitions of the form (4) or (5), are present in δ where they enter state q_0. Since M is reversible, the modified transitions do not violate the reversibility either. We conclude that M' satisfies the required properties, is equivalent to M, and is of the same type as M. □

To conclude the consideration of stationary transitions we present the result that every length-preserving reversible DPDT can be transformed into an equivalent reversible one that moves at least one head in every but possibly the last step of a computation. The basic idea of the proof of the next proposition is to simulate a possibly empty sequence of stationary transitions and one following non-stationary transition at once.

Proposition 4 *Every length-preserving reversible DPDT can be converted into an equivalent length-preserving reversible DPDT of the same type that always halts and moves its input head or emits an output symbol in every but possibly the last step of a computation.*

Proof. Let $M = \langle Q, \Sigma, \Gamma, \Delta, \lhd, \bot, \sqcup, \delta, q_0, F\rangle$ be a REV-DPDT. By Lemma 3 we may assume that M performs at most m consecutive stationary transitions in any halting computation, where m is a constant, and that never enters an initial configuration again.

Since M is reversible, any state is reached solely by stationary transitions or solely by non-stationary transitions. To figure out which case applies to a certain state it is sufficient to inspect the transition function. Moreover, to check whether a sequence of stationary transitions starting from a given configuration is finite, we have to simulate at most $m + 1$ steps of M.

In order to construct an equivalent length-preserving reversible DPDT of the same type with the properties claimed, basically, the idea is to simulate a possibly empty sequence of stationary transitions and one following non-stationary

transition at once. If M runs into an infinite loop of stationary transitions or performs stationary transitions on the endmarker, then the sequence of stationary transitions is simulated without a following non-stationary transition.

The DPDT $M' = \langle Q', \Sigma, \Gamma', \Delta, \lhd, \bot, \sqcup, \delta', q_0', F' \rangle$ is constructed as follows. Recall that M' simulates at most some $m+1$ steps of M at once during which it has access to no more than the topmost $m+1$ pushdown symbols. On the other hand, it cannot push more than $m+1$ symbols onto the store. So, we add a register to the states in which M' can store up to $2m$ pushdown symbols of M (the topmost ones), and consider every string of $m+1$ pushdown symbols of M to be a single pushdown symbol of M'. Let $F_+ = \{ q_+ \mid q \in F \}$ be a copy of F. Then define

$$\begin{array}{rclcrcl} Q' & = & (Q \cup F_+) \times \Gamma^{\leq 2m}, & & \Gamma' & = & \Gamma^{m+1}, \\ q_0' & = & (q_0, \lambda), & & F' & = & (F \cup F_+) \times \Gamma^{\leq 2m}. \end{array}$$

Given $(p, x_i x_{i-1} \cdots x_1) \in Q \times \Gamma^{\leq 2m}$, $a \in \Sigma_\lhd$, and $z_{m+1} z_m \cdots z_1 \in \Gamma'$, the transition $\delta'((p, x_i x_{i-1} \cdots x_1), a, z_{m+1} z_m \cdots z_1)$ is defined if and only if p is reached by non-stationary transitions in M. In this case, it is defined by the computation of M starting in configuration $c_0 = (\lambda, p, a, x_i x_{i-1} \cdots x_1 z_{m+1} z_m \cdots z_1, \lambda)$.

First assume that the computation starts with a possibly empty sequence of stationary transitions followed by a non-stationary transition. That is, the computation is $c_0 \vdash c_1 \vdash \cdots \vdash c_n$, where $n \leq m+1$. We distinguish two cases: If the last step moves the input head we have $c_n = (a, q, \lambda, y_j y_{j-1} \cdots y_1, d)$ and define $\delta'((p, x_i x_{i-1} \cdots x_1), a, z_{m+1} z_m \cdots z_1) = ((q, \gamma), \mathtt{op}, d', 1)$, where $d' = \sqcup$ if $d = \lambda$ and $d' = d$ otherwise, and

$$\begin{array}{lll} \gamma = y_j y_{j-1} \cdots y_1, & \mathtt{op} = \mathtt{pop} & \text{if } 0 \leq j \leq 2m, \\ \gamma = y_j y_{j-1} \cdots y_{m+2}, & \mathtt{op} = \mathtt{top} & \text{if } 2m+1 \leq j \leq 3m+1, \\ \gamma = y_j y_{j-1} \cdots y_{2m+3}, & \mathtt{op} = \mathtt{push}(y_{2m+2} \cdots y_{m+2}) & \text{if } 3m+2 \leq j \leq 4m+2. \end{array}$$

Note that in the second and third alternatives, M could not modify the symbols $z_{m+1} z_m \cdots z_1$ and, therefore, $y_{m+1} y_m \cdots y_1 = z_{m+1} z_m \cdots z_1$.

The second case is similar. If the last step emits an output symbol without moving the input head we have $c_n = (\lambda, q, a, y_j y_{j-1} \cdots y_1, d)$ with $d \in \Delta$ and define $\delta'((p, x_i x_{i-1} \cdots x_1), a, z_{m+1} z_m \cdots z_1) = ((q, \gamma), \mathtt{op}, d, 0)$, where γ and $\mathtt{op}$ are as for the first case.

Next, assume that the computation of M starts with a sequence of stationary transitions and halts without performing a non-stationary transition. Then we have $c_n = (\lambda, q, a, y_j y_{j-1} \cdots y_1, \lambda)$ and $\delta'((p, x_i x_{i-1} \cdots x_1), a, z_{m+1} z_m \cdots z_1)$ remains undefined for $a \neq \lhd$ or $q \notin F$. If $a = \lhd$ and $q \in F$ we define the transition $\delta'((p, x_i x_{i-1} \cdots x_1), \lhd, z_{m+1} z_m \cdots z_1) = ((q_+, x_i x_{i-1} \cdots x_1), \mathtt{top}, \sqcup, 0)$.

Finally, if the computation of M is an infinite sequence of stationary transitions then $\delta'((p, x_i x_{i-1} \cdots x_1), a, z_{m+1} z_m \cdots z_1)$ remains undefined. Moreover, the transition function δ' remains undefined for states whose first component belongs to F_+.

If the initial state q_0 of M is reached by non-stationary moves, the initial configuration of M' does not need to be treated in any special way. However,

if q_0 is reached by stationary moves, we proceed as follows. First, all transitions of δ' defined so far are checked whether they are based on a computation $c_0 \vdash c_1 \vdash \cdots \vdash c_n$ of M that includes an initial configuration. If yes, the transition is undefined again. This can safely be done since by assumption the initial configuration only appears at the beginning of any computation of M. Next, the transition $\delta'((q_0, \lambda), a, \bot)$ is defined as above, for any input symbol a.

Finally, the completion of the definition of δ' for the situations in which the bottom-of-pushdown symbol is the topmost symbol is straightforward. This concludes the construction of M'.

Given an input w, the computation of M is unambiguously split into sequences of steps each of which is performed by M' at once. If M accepts, so does M'. Conversely, every step of M' corresponds to a sequence of steps of M. So, we have $T(M) = T(M')$.

To give evidence that M' is reversible we argue with configurations. Except for possibly the initial configuration and halting configurations on the end-marker, any configuration of M' includes a state of M that is reachable with non-stationary moves. The unique predecessor configuration is determined by running M backwards until either the initial configuration appears or another configuration with a state of M that is reachable with non-stationary moves is reached. So, the reversibility of M' follows from the reversibility of M. Since by assumption the initial configuration of M never occurs again, also the steps involving the initial configuration of M' are reversible. Finally, also the configurations including states from F_+ have a unique predecessor that can be determined by running M backwards as above. □

In particular, the normalization result eliminates stationary moves from reversible computations. This implies the first comparison between families under consideration.

Theorem 5 *The families $\mathscr{T}$(REV-M-DPDT) and $\mathscr{T}$(REV-s-DPDT) coincide.*

Proof. The inclusion $\mathscr{T}$(REV-M-DPDT) $\subseteq$ $\mathscr{T}$(REV-s-DPDT) follows from the definition. Conversely, applying Proposition 4 to a given REV-s-DPDT yields a REV-M-DPDT. We conclude $\mathscr{T}$(REV-M-DPDT) $\supseteq$ $\mathscr{T}$(REV-s-DPDT) and, thus, $\mathscr{T}$(REV-M-DPDT) = $\mathscr{T}$(REV-s-DPDT). □

However, the equivalence of both transduction families gets lost if irreversible computations are allowed.

Theorem 6 *There is a strict inclusion between the families $\mathscr{T}$(M-DPDT) and $\mathscr{T}$(s-DPDT).*

Proof. The inclusion $\mathscr{T}$(M-DPDT) $\subseteq$ $\mathscr{T}$(s-DPDT) follows from the definition. To define the transduction that witnesses the properness of the inclusion we first consider the function $f : \{a, b\}^+ \times \{a, b\}^+ \to \{a, b, c\}^+$ which is defined by

$$f(x_1 x_2 \cdots x_n, y_1 y_2 \cdots y_m) = \begin{cases} x_n x_{n-1} \cdots x_{n-m+1} & \text{if } m < n, \\ x_n x_{n-1} \cdots x_1 c^{m-n} & \text{if } m \geq n. \end{cases}$$

We note that $|f(x,y)| = |y|$ for all $x, y \in \{a,b\}^+$. Then, consider transductions

$$\begin{aligned} \tau_1 &= \{\, (u\#v\$_1 w, u\#v\$_1 f(u,w)) \mid u,v,w \in \{a,b\}^+ \,\}, \\ \tau_2 &= \{\, (u\#v\$_2 w, u\#v\$_2 f(v,w)) \mid u,v,w \in \{a,b\}^+ \,\}, \end{aligned}$$

and their union $\tau = \tau_1 \cup \tau_2$. We now claim that τ can be realized by an s-DPDT, but not by any M-DPDT. For the first claim we now describe an s-DPDT M for τ. Basically, M reads and emits every symbol of the prefix $u\#v$ up to the symbol $\$_1$ or $\$_2$. Additionally, $u\#v$ is pushed on the pushdown store. In case of a separating symbol $\$_1$, M remains on $\$_1$ and removes $v^R\#$ from the pushdown store. Then, the remaining input is read and for every symbol read one symbol from u^R is popped from the pushdown store and emitted. If the remaining input is shorter than u^R then only a prefix of u^R is emitted, otherwise additional symbols c are emitted if needed. In case of a separating symbol $\$_2$, the remaining input is read and for every symbol read one symbol from v^R is popped from the pushdown store and emitted. Again, if the remaining input is shorter than v^R then only a prefix of v^R is emitted, otherwise additional symbols c are emitted if needed. Clearly, M is an s-DPDT that realizes τ.

Now, we assume that τ is realized by an M-DPDT M'. Since M' reads and emits a symbol in every step, it is possible to consider M' as a realtime DPDA A that works over an alphabet of paired symbols where the left symbol is the input read and the right symbol is the output written. Hence, A accepts the language $\{\, (x_1,y_1)(x_2,y_2)\cdots(x_n,y_n)\lhd \mid (x_1x_2\cdots x_n, y_1y_2\cdots y_n) \in T(M) \,\}$ in real time. Next, we intersect $L(A)$ with the regular language

$$(a,a)^+(\#,\#)(a,a)^+\{(\$_1,\$_1),(\$_2,\$_2)\}(a,a)^+(a,c)\lhd$$

and obtain that language

$$\begin{aligned} L = \{\, (a,a)^n(\#,\#)(a,a)^m(\$_1,\$_1)(a,a)^n(a,c)\lhd \mid m,n \geq 1 \,\} \cup \\ \{\, (a,a)^n(\#,\#)(a,a)^m(\$_2,\$_2)(a,a)^m(a,c)\lhd \mid m,n \geq 1 \,\} \end{aligned}$$

is realtime deterministic context free. However, it is shown in [22] by using a particular pumping lemma for realtime deterministic context-free languages that $\{\, a^nb^ma^n \mid n,m \geq 1 \,\} \cup \{\, a^nb^mc^m \mid n,m \geq 1 \,\}$ is not realtime deterministic context free. A similar argument can be applied to L and shows that $L(A)$ is not realtime deterministic context free which gives a contradiction and concludes the proof of the theorem. □

We would like to note that the input part of the witness transduction τ used in the above proof is a regular language which is clearly accepted by DPDAs as well as by realtime DPDAs. Thus, the separation of the families $\mathscr{T}$(M-DPDT) and $\mathscr{T}$(s-DPDT) relies in fact on the power of the different types of pushdown transducers and not on the different computational power of the underlying pushdown automata.

3.2. Bounded Delay versus Strongly Length-Preserving

We turn to examine the computational capacity gained in the possibility of having the input head and output head asynchronous but not too far from each other. Their distance has to be bounded by a constant. For any alphabet Σ with $a \in \Sigma$ and such that $|\Sigma| \geq 2$, we consider the witness transduction $\textsc{lRotate} = \{ (aw, wa) \mid w \in \Sigma^* \}$.

Proposition 7 *The transduction* lRotate *is realized by a REV-bd-DPDT.*

Proof. We first informally describe a bd-DPDT T realizing lRotate. Actually T does not use the pushdown, it is basically the identity transducer (one state with self-loops on every input letter copying this letter to the output) to which a preliminary step (from the initial state) and a final step (to the unique final state) are added: from the initial state reading the input letter a, the transducer enters the central looping state without producing an output symbol and moves its input head; symmetrically, from the central looping state reading the endmarker, the transducer enters the unique final state and emits the output symbol a.

Next, we transform T into a REV-bd-DPDT without changing the delay. First, the self-loops are divided into two steps, in order to satisfy the head move constraints required by the definition of reversibility (1). Second, we use the pushdown to save information about the history of the computation, namely, the pushdown contains the number of times the loops have been taken (in unary). In this way, in a backward computation, the device is able to detect the first time the central strongly connected component was entered.

Formally, we define the REV-bd-DPDT T' as follows (see Figure 2):

$$T' = \langle \{q, s, t, f\}, \Sigma, \{a\}, \Delta, \lhd, \bot, \sqcup, \delta, q, \{f\} \rangle$$

where the alphabets Σ and Δ are equal and contain the symbol a, and, for all $\sigma \in \Sigma$, the transition function δ is defined by:

$$\begin{array}{rcccl} & & \delta(q, a, \bot) & = & (s, \mathtt{top}, \sqcup, 1) \\ \delta(s, \sigma, a) & = & \delta(s, \sigma, \bot) & = & (t, \mathtt{top}, \sigma, 0) \\ \delta(t, \sigma, a) & = & \delta(t, \sigma, \bot) & = & (s, \mathtt{push}(a), \sqcup, 1) \\ \delta(s, \lhd, a) & = & \delta(s, \lhd, \bot) & = & (f, \mathtt{top}, a, 0) \end{array}$$

□

The next result yields the separation. The transduction lRotate is not even realized by an irreversible s-DPDT.

Proposition 8 *The transduction* lRotate *cannot be realized by any s-DPDT.*

Proof. By contradiction, assume that lRotate is realized by some s-DPDT T. Suppose that in the successful computation of T on input ab, the first non-stationary step is $\delta(p, a, g) = (q, \mathtt{op}, d, 1)$, for some states p and q, $g \in \Gamma_\bot$,

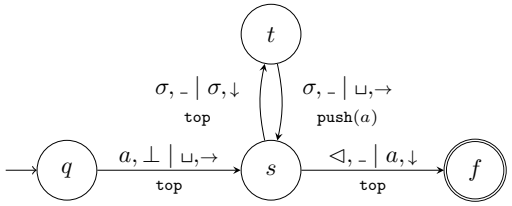

Figure 2: A REV-bd-DPDT realizing the transduction lRotate. The symbol σ represents any symbol of the input alphabet.

and $d \in \Delta_{\sqcup}$. Since T is strongly length-preserving, we have $d \neq \sqcup$ and furthermore d is the first symbol emitted. So, $d = b$, since the output associated to ab is ba. Since T is deterministic, a computation on input aa will have the same first non-stationary step and, therefore, a b is emitted during the computation. This is a contradiction since the output aa associated to the input aa does not contain the symbol b. □

Since the two inclusions $\mathscr{T}$(REV-s-DPDT) $\subseteq$ $\mathscr{T}$(REV-bd-DPDT) and $\mathscr{T}$(s-DPDT) $\subseteq$ $\mathscr{T}$(bd-DPDT) follow from the definitions, Propositions 7 and 8 imply the following separations.

Theorem 9 *The family $\mathscr{T}$(REV-bd-DPDT) strictly includes the family $\mathscr{T}$(REV-s-DPDT), and the family $\mathscr{T}$(bd-DPDT) strictly includes the family $\mathscr{T}$(s-DPDT).*

3.3. Weakly Length-Preserving versus Bounded Delay

The most powerful variant of length-preserving transducers works weakly length-preserving, where now the distance between input head and output head may be arbitrary. We first show this yields strictly more computational capacity. To this end, we use the transduction mSwap $= \{ (a^m\#b^n, b^n\#a^m) \mid m, n \geq 0 \}$.

Proposition 10 *The transduction* mSwap *is realized by a REV-w-DPDT.*

Proof. The transduction mSwap is realized by the REV-w-DPDT M depicted in Figure 3 whose state set is $\{q, s, t, f\}$, where q is the initial state and f is the sole final state. The transducer M uses only one pushdown symbol, denoted a, meaning that it can be viewed as a *counter transducer*. The transition function δ of M is defined as follows:

$$\begin{array}{rcrcl}
\delta(q, a, \bot) & = & \delta(q, a, a) & = & (q, \mathtt{push}(a), \sqcup, 1) \\
\delta(q, \#, \bot) & = & \delta(q, \#, a) & = & (s, \mathtt{top}, \sqcup, 1) \\
\delta(s, b, \bot) & = & \delta(s, b, a) & = & (t, \mathtt{top}, b, 0) \\
\delta(t, b, \bot) & = & \delta(t, b, a) & = & (s, \mathtt{top}, \sqcup, 1) \\
\delta(s, \lhd, \bot) & = & \delta(s, \lhd, a) & = & (f, \mathtt{top}, \#, 0) \\
 & & \delta(f, \lhd, a) & = & (f, \mathtt{pop}, a, 0)
\end{array}$$

So defined, the transducer M realizes mSwap. In fact, in any successful computation on some input word $a^m\#b^n$, $m, n \geq 0$, when the transducer performs

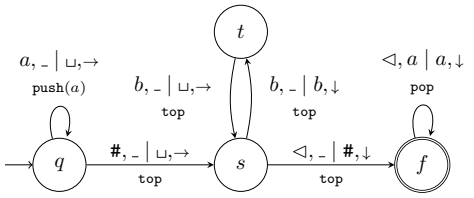

Figure 3: A REV-w-DPDT realizing mSwap.

the transition from q to s, the configuration becomes $(a^m\#, s, b^n\triangleleft, a^m\bot, \lambda)$. Similarly, when it performs the transition from s to f, the configuration becomes $(a^m\#b^n, f, \triangleleft, a^m\bot, b^n\#)$. Finally, since transducers are required to halt in order to accept, M has to perform the self-loop around f until having emptied the pushdown. Hence, when it halts, the configuration is $(a^m\#b^n, f, \triangleleft, \bot, b^n\#a^m)$. For every such input word, the successful computation described above exists. Thus, R realizes the transduction mSwap. Moreover, it is immediately verified that R is reversible. □

The next result yields the separation: the transduction mSwap is not even realized by an irreversible bd-DPDT.

Proposition 11 *The transduction* mSwap *is not realizable by any bd-DPDT.*

Proof. By contradiction, suppose that a bd-DPDT T realizes mSwap, and let the delay be bounded by some $k \geq 0$. Consider the input words in $a^{k+1}\#b^*$, which are all valid inputs for the transduction. On each of these words, by the determinism of T, when the input head reaches the $(k+1)$st symbol a at least one symbol of the output has been written (otherwise, the output head is more than k cells delayed). If the first output letter produced so far is a or #, we get a contradiction for the input $a^{k+1}\#b$ whose associated output begins by b. If it is b, then we get a contradiction for the input word $a^{k+1}\#$, whose associated output begins by #. Hence, no bd-DPDT realizes mSwap. □

Since the two inclusions $\mathscr{T}$(REV-bd-DPDT) $\subseteq$ $\mathscr{T}$(REV-w-DPDT) and $\mathscr{T}$(bd-DPDT) $\subseteq$ $\mathscr{T}$(w-DPDT) follow from the definitions, Propositions 10 and 11 imply the following separations.

Theorem 12 *The family $\mathscr{T}$(REV-w-DPDT) strictly includes the family $\mathscr{T}$(REV-bd-DPDT), and the family $\mathscr{T}$(w-DPDT) strictly includes the family $\mathscr{T}$(bd-DPDT).*

3.4. Irreversibility versus Reversibility

In the previous sections we have established the finite hierarchies dependent on the modes of transductions for reversible as well as general devices. Here we compare the computational capacities of general and reversible classes with each other. A witness transduction is

$$\text{aOccur} = \{\, (a^n\#w\#a^n, a^n\#a^{|w|}\#a^n) \mid n \geq 0, w \in \{a,b\}^+, |w|_a \geq 1 \,\}.$$

It is not hard to see that aOccur is realized even by the weakest not necessarily reversible device in question, that is, by some M-DPDT. However, even reversible pushdown transducers with bounded delay cannot realize it.

Proposition 13 *The transduction* aOccur *cannot be realized by any reversible bounded-delay pushdown transducer.*

Proof. Assume that a REV-bd-DPDT $M = \langle Q, \Sigma, \Gamma, \Delta, \lhd, \bot, \sqcup, \delta, q_0, F\rangle$ realizes aOccur, and let the delay be bounded by some $k \geq 0$. By Proposition 4 we may safely assume that M always halts and never performs a stationary move in every but possibly the last step of a computation.

First, we look at the initial parts of the computations on input prefixes a^n, where n is large enough, and we want to show that the pushdown contents have to be different after the processing of infinitely many different prefixes a^n. By contradiction, we assume that there are only a finite number of pushdown contents that appear after processing the prefixes a^n, for all $n \geq 0$. Since the delay is bounded, there are only finitely many possible delays. The number of states is finite as well. So, there must be two numbers n_1 and n_2 such that the delay of the output head, the current state, and the pushdown contents are identical after processing a^{n_1} and a^{n_2}. Then, for the suffix $\#a\#a^{n_1}$ both computations would continue successfully. From this contradiction we conclude that the pushdown contents have to be different after the processing of infinitely many different prefixes a^n.

We conclude that the pushdown height grows with n. Therefore, M eventually runs through cycles that increase the pushdown height, say,

$$\begin{aligned}(\lambda, q_0, a^n\#, \bot, \lambda) &\vdash^* (a^i, q, a^{n-i}\#, \gamma_0, \beta_0)\\ &\vdash^* (a^{i+j}, q, a^{n-i-j}\#, \gamma_1\gamma_0, \beta_0\beta_1) \vdash^* (a^{i+2j}, q, a^{n-i-2j}\#, \gamma_1\gamma_1\gamma_0, \beta_0\beta_1\beta_1),\end{aligned}$$

for $j > 0$ and $\gamma_1 \neq \lambda$. Here, we may safely assume that M moves its input head when it enters the state q. On prefixes a^n with n from the set $N = \{\, i + xj \mid x \geq 0 \,\}$ the transducer M runs through complete cycles.

Second, we look at the computations on the input factors between the # symbols. To this end, we choose some fixed number $m \geq k+1$, only numbers n from the set N, and let $\ell \geq 1$ be large enough. Then we consider factors $(b^m a)^\ell$ whose processing starts in configurations $(a^n, q, \#(b^m a)^\ell, \gamma, \beta)$ where γ is of the form $\gamma_1^*\gamma_0$. Since the computations are on cyclic input factors $(b^m a)^*$ and start with cyclic topmost pushdown contents γ_1^*, the computations themselves will be cyclic. We can always unroll the cycle such that it matches the $b^m a$ blocks. That is,

$$\begin{aligned}(a^n, q, \#(b^m a)^\ell, \gamma, \beta) &\vdash^* (a^n\#(b^m a)^i, p, (b^m a)^{\ell-i}, \gamma_0', \beta_0')\\ &\vdash^* (a^n\#(b^m a)^{i+j}, p, (b^m a)^{\ell-i-j}, \gamma_1', \beta_1')\\ &\vdash^* (a^n\#(b^m a)^{i+2j}, p, (b^m a)^{\ell-i-2j}, \gamma_2', \beta_2'),\end{aligned}$$

where β_0', β_1', and β_2' are of the form $a^n\#a^+$.

We now want to obtain that the pushdown height during a cycle on the input factor between the # symbols remains constant. To this end, we show by way of contradiction that the pushdown height neither decreases nor increases during such a cycle. For the first case, we assume that the pushdown height decreases during such a cycle, that is, $|\gamma_2'| < |\gamma_1'| < |\gamma_0'|$ and $\gamma_1' = \hat{\gamma}\gamma_1^x\gamma_0$, $\gamma_2' = \hat{\gamma}\gamma_1^y\gamma_0$ with $y < x$. Then there are two numbers $n_1 < n_2$ from the set N, two numbers $r_1 < r_2$, and two numbers $s_1, s_2 \geq 1$ such that

$$(a^{n_1}, q, \#(b^m a)^\ell, \gamma, \beta) \vdash^* (a^{n_1}\#(b^m a)^{i+r_1 j}, p, (b^m a)^{\ell-i-r_1 j}, \hat{\gamma}\gamma_1^x\gamma_0, a^{n_1}\#a^{s_1})$$

and

$$(a^{n_2}, q, \#(b^m a)^\ell, \bar{\gamma}, \bar{\beta}) \vdash^* (a^{n_2}\#(b^m a)^{i+r_2 j}, p, (b^m a)^{\ell-i-r_2 j}, \hat{\gamma}\gamma_1^x\gamma_0, a^{n_2}\#a^{s_2}).$$

This implies the halting computation

$$\begin{aligned}(a^{n_1}, q, \#(b^m a)^{\ell-(r_2-r_1)j}\#a^{n_2}\lhd, \gamma, \beta)\\ \vdash^* (a^{n_1}\#(b^m a)^{i+r_1 j}, p, (b^m a)^{\ell-i-r_2 j}\#a^{n_2}\lhd, \hat{\gamma}\gamma_1^x\gamma_0, a^{n_1}\#a^{s_1})\\ \vdash^* (a^{n_1}\#(b^m a)^{\ell-(r_2-r_1)j}\#a^{n_2}, p_+, \lhd, \gamma_f, a^{n_1}\#a^s\#a^{n_2})\end{aligned}$$

for some accepting state p_+ and some $s \geq 1$. We conclude from the contradiction that the pushdown height does not decrease during a cycle on the input factor between the # symbols.

For the second case, we assume that the pushdown height increases during such a cycle, that is, $|\gamma_2'| > |\gamma_1'| > |\gamma_0'|$ and $\gamma_1' = \gamma_2\hat{\gamma}\gamma_1^x\gamma_0$, $\gamma_2' = \gamma_2^{1+z}\hat{\gamma}\gamma_1^y\gamma_0$ with $z \geq 1$. Then there are two numbers $n_1 < n_2$ from the set N (recall that ℓ is large enough), a number $r \geq 0$, two numbers $x_1, x_2 \geq 1$, and two numbers $s_1, s_2 \geq 1$ such that

$$(a^{n_1}, q, \#(b^m a)^\ell, \gamma, \beta) \vdash^* (a^{n_1}\#(b^m a)^{i+rj}, p, (b^m a)^{\ell-i-rj}, \gamma_2^t\hat{\gamma}\gamma_1^{x_1}\gamma_0, a^{n_1}\#a^{s_1})$$

and

$$(a^{n_2}, q, \#(b^m a)^\ell, \bar{\gamma}, \bar{\beta}) \vdash^* (a^{n_2}\#(b^m a)^{i+rj}, p, (b^m a)^{\ell-i-rj}, \gamma_2^t\hat{\gamma}\gamma_1^{x_2}\gamma_0, a^{n_2}\#a^{s_2}),$$

where we safely may assume $t > 2(n_2 + 2) + k$, since r can be large enough since ℓ is large enough. Since M does not perform stationary moves, on input $a^{n_1}\#(b^m a)^{i+rj}\#a^{n_2}$ there are at most $2(n_2+2)+k$ steps left until the end of the computation. During these steps at most the topmost t pushdown symbols can be accessed. So, as before, we derive a contradiction since there is a successful computation with prefix $a^{n_1}\#$ and suffix $\#a^{n_2}$. We conclude that the pushdown height also does not increase during a cycle on the input factor between the # symbols, that is, it remains constant in a cycle.

So, we have $\gamma_1' = \tilde{\gamma}\gamma_1^x\gamma_0 = \gamma_2'$. Finally, we consider the configurations when M moves its input head on an a in the factor between the # symbols.

Eventually, this happens within the cycle:

$$\begin{aligned}(a^n\#b^m, p_1, a(b^ma)^{\ell-1}, \gamma, \beta) &\vdash^* (a^n\#(b^ma)^ib^m, p_2, a(b^ma)^{\ell-i-1}, \tilde{\gamma}_0, \tilde{\beta}_0)\\ &\vdash^* (a^n\#(b^ma)^{i+j}b^m, p_2, a(b^ma)^{\ell-i-j-1}, \tilde{\gamma}_0, \tilde{\beta}_1)\\ &\vdash^* (a^n\#(b^ma)^{i+ij}b^m, p_2, a(b^ma)^{\ell-i-ij-1}, \tilde{\gamma}_0, \tilde{\beta}_2).\end{aligned}$$

Since M is reversible, from the second configuration we uniquely can run it backwards for $i(m+1)$ steps in which the input head is moved. This yields the configuration $(a^n\#b^m, p_1, a(b^ma)^{\ell-1}, \gamma, \beta)$. Since M is of bounded delay, the suffixes of $\tilde{\beta}_0$ and $\tilde{\beta}_2$ are of the form a^+ and are long enough. Therefore, running M backwards for $i(m+1)$ steps in which the input head is moved from the fourth configuration, must be the same backwards computation as running backwards from the second configuration. So, running backwards from the fourth configuration yields $(a^n\#(b^ma)^{ij}b^m, p_1, a(b^ma)^{\ell-ij-1}, \gamma, \tilde{\beta}_3)$ and, thus, p_1 appears in the cycle. Since $a^n\#(b^ma)^{ij}b^m\#a^n$ is successfully transduced so is $a^n\#b^m\#a^n$, a contradiction. We conclude that the transduction aOccur is not realized by any reversible bounded-delay pushdown transducer. □

Since any reversible transducer is a special case of a general one of the same type and the transduction aOccur is realized by an M-DPDT, we have the following inclusions.

Theorem 14 *The family $\mathscr{T}$(bd-DPDT) strictly includes the family $\mathscr{T}$(REV-bd-DPDT), and the families $\mathscr{T}$(s-DPDT) and $\mathscr{T}$(M-DPDT) strictly include the family $\mathscr{T}$(REV-s-DPDT) = $\mathscr{T}$(REV-M-DPDT).*

To compare the transducers working in weak mode, we use a variant of the transduction mSwap without center marker, that is, we define the transduction $\textsc{Swap} = \{\, (a^mb^n, b^na^m) \mid m, n \geq 0 \,\}$.

Proposition 15 *The transduction* Swap *is realized by a w-DPDT.*

Proof. A w-DPDT realizing Swap starts to read the leading a's and stores them onto the pushdown without emitting symbols. Then it reads and emits the following b's without changing the pushdown content. Finally, on the endmarker it emits the a's from the pushdown. A precise construction is shown in Figure 4. □

We can show that the transduction swap is not realizable by a reversible transducer.

Proposition 16 *The transduction* swap *cannot be realized by any reversible pushdown transducer.*

Proof. In contrast to the assertion, assume that there exists a REV-w-DPDT $M = \langle Q, \Sigma, \Gamma, \Delta, \lhd, \bot, \sqcup, \delta, q_0, F\rangle$ which realizes Swap. By Proposition 4 we may safely assume that M always halts and never performs a stationary move

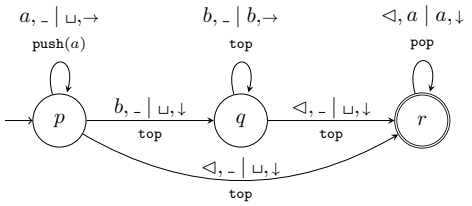

Figure 4: A w-DPDT realizing the transduction Swap.

in every but possibly the last step of a computation. Later we modify M and have to preserve reversibility. In order to simplify matters concerning this possibly last step on the endmarker, we do the following. Let F' be the primed copy of F. Then we add F' to the state set of M and exchange F by F'. Whenever $\delta(p, \triangleleft, g)$ is undefined, state p belongs to F, and state p is reached by a non-stationary transition, we define $\delta(p, \triangleleft, g) = (p', \mathtt{top}, \sqcup, 0)$. Whenever $\delta(q, \triangleleft, g) = (p, \mathtt{op}, \sqcup, 0)$ is defined and state p belongs to F, we redefine $\delta(q, \triangleleft, g) = (p', \mathtt{op}, \sqcup, 0)$. In the former case we have added one stationary transition that becomes the last transition in a successful computation which previously ended with a non-stationary step. In the latter case we have replaced the stationary last step in a successful computation by a stationary transition that leads to a primed copy of the accepting state. Since there are no outgoing transitions from the new final states from F', this modification neither violates the reversibility nor the transduction realized.

After this preparation, we observe that the transitions performed during any successful computation of M satisfy:

$$\delta(p, a, g) = (q, \mathtt{op}, d, m) \implies d = \sqcup \tag{1}$$

$$\delta(p, b, g) = (q, \mathtt{op}, d, m) \implies d \neq a \tag{2}$$

$$\delta(p, x, g) = (q, \mathtt{op}, a, m) \implies x = \triangleleft \tag{3}$$

where p, q are states, $g \in \Gamma_\perp$, $m \in \{0, 1\}$, and $d \in \{a, b\}_\sqcup$. Indeed, if M would not satisfy (1) then it would fail, since it might produce a symbol before having checked the existence of a symbol b in the suffix of the input. Similarly, M would fail if it does not satisfy (2) because it might emit a symbol a before having completely emitted the b's, which should form a prefix of the output. Lastly, (3) is implied by (1) and (2).

Next, we turn to show how to obtain a reversible pushdown automaton A that accepts the language $\{ a^n b^n \mid n \geq 0 \}$ from M. Since no reversible pushdown automaton can recognize this language [10, Lemma 6], we obtain a contradiction that shows the theorem.

The first step of the construction of A is to modify M such that it only accepts inputs from a^* while preserving all the properties seen above as well as reversibility. To this end, it is sufficient to remove every transition reading or emitting a b from δ. By the properties above there is no transition that reads an a and emits a b or vice versa. So, this construction step can safely be done

without changing anything for the computations on inputs from a^*. Let M' be the transducer obtained in this way. It is reversible, since we only removed transitions from M. Moreover, its transitions still satisfy the three conditions given above.

In particular, on input a^* transducer M' first reads the a's without emitting anything. When the endmarker is reached, M' emits the a's read and halts in an accepting state from F. The next step is to construct the reversible pushdown automaton $A = \langle Q, \{a, b\}, \Gamma, \triangleleft, \bot, \delta', q_0, F\rangle$ from M'. The transition function δ' is defined as follows:

$$\begin{array}{rclcrcl} \delta'(p,a,g) & = & (q,\mathtt{op},1) & \iff & \delta(p,a,g) & = & (q,\mathtt{op},\sqcup,1) \\ \delta'(p,b,g) & = & (q,\mathtt{op},1) & \iff & \delta(p,\triangleleft,g) & = & (q,\mathtt{op},a,0) \\ \delta'(p,\triangleleft,g) & = & (q,\mathtt{op},0) & \iff & \delta(p,\triangleleft,g) & = & (q,\mathtt{op},\sqcup,0) \end{array}$$

for each $p, q \in Q$, $g \in \Gamma_\bot$, and $\mathtt{op} \in \{\mathtt{push}(x) \mid x \in \Gamma\} \cup \{\mathtt{top}, \mathtt{pop}\}$.

In this way, A simulates M', but whenever M' emits an a while reading the endmarker, A reads a b. So, for each successful transduction of M' on an input a^n, we obtain an accepting computation of A on input $a^n b^n$, and vice versa. Hence, A accepts the language $\{\, a^n b^n \mid n \geq 0 \,\}$.

Finally, it is not hard to verify that A is reversible since M' is reversible and the construction of δ' does not violate reversibility.

It is worth mentioning that A moves its input head in any but a possible last step on the endmarker. The pushdown automata considered in [10] do not have an input endmarker. However, the proof that the language $\{\, a^n b^n \mid n \geq 0 \,\}$ is not accepted by any reversible pushdown automaton does not rely on the endmarker. It literally applies in this case as well. □

So, we have shown the following theorem.

Theorem 17 *The family $\mathscr{T}$(w-DPDT) strictly includes $\mathscr{T}$(REV-w-DPDT).*

The next goal is to develop a witness transduction that shows the missing incomparability results, that is, it should be realizable by some M-DPDT but not by any REV-w-DPDT. To this end, let $L_{bin} = ((aa+a)(bb+b))^*(aa+a+\lambda)$ be a regular language and define the transduction

$$3\textsc{Occur} = \{\, (a^n \# w \# a^n, a^n \# a^{|w|} \# a^n) \mid n \geq 0, w \in L_{bin} \,\}.$$

As for the transduction aOccur it is not hard to see that 3Occur is realized by the weakest not necessarily reversible device in question, that is, by some M-DPDT. In order to show that 3Occur cannot be realized by any REV-w-DPDT we will use Kolmogorov complexity and incompressibility arguments. General information on Kolmogorov complexity and the incompressibility method can be found, for example, in [23]. Let $w \in \{0,1\}^+$ be an arbitrary binary string. The Kolmogorov complexity $C(w)$ of w is defined to be the minimal size of a binary program describing w. The following key argument for the incompressibility method is well known. There are binary strings w of *any* length so that $|w| \leq C(w)$.

Next, we encode words $w \in \{0,1\}^+$ as follows. From left to right the digits are represented alternating by a's and b's so that a 0 is represented by a single letter and a 1 by a double letter. For example, the word 010110 is encoded as *abbabbaab*. Let $t(w)$ denote the code of w. Clearly, for any word $w \in \{0,1\}^+$, its code $t(w)$ belongs to the regular language L_{bin}.

Proposition 18 *There exists no REV-w-DPDT that realizes the transduction* 3OCCUR.

PROOF. In contrast to the assertion, assume that there exists a REV-w-DPDT $M = \langle Q, \Sigma, \Gamma, \Delta, \lhd, \bot, \sqcup, \delta, q_0, F\rangle$ which realizes 3OCCUR. By Proposition 4 we may safely assume that M always halts and never performs a stationary move in every but possibly the last step of a computation.

First, we consider successful computations of M on input prefixes a^n, where n is large enough. On these input prefixes, no combination of state and pushdown content may appear twice. If otherwise

$$\begin{aligned}(\lambda, q_0, a^n\#w\#a^n\lhd, \bot, \lambda) &\vdash^* (a^{m_1}, q_1, a^{n-m_1}\#w\#a^n\lhd, \gamma_1, \beta_1)\\ &\vdash^+ (a^{m_1+m_2}, q_1, a^{n-m_1-m_2}\#w\#a^n\lhd, \gamma_1, \beta_2)\end{aligned}$$

is the beginning of a successful computation, then so is

$$(\lambda, q_0, a^{n-m_2}\#w\#a^n\lhd, \bot, \lambda) \vdash^* (a^{m_1}, q_1, a^{n-m_1-m_2}\#w\#a^n\lhd, \gamma_1, \beta_1),$$

but there is no pair in 3OCCUR whose first component is $a^{n-m_2}\#w\#a^n$, a contradiction.

We conclude that the pushdown height grows with n. Therefore, M eventually runs through cycles that increase the pushdown height, say,

$$\begin{aligned}&(\lambda, q_0, a^n\#, \bot, \lambda) \vdash^* (a^{t_1}, q_1, a^{n-t_1}\#, \gamma_1, \beta_1)\\ &\vdash^+ (a^{t_1+c_1}, q_1, a^{n-t_1-c_1}\#, \gamma_2\gamma_1, \beta_2) \vdash^+ (a^{t_1+2c_1}, q_1, a^{n-t_1-2c_1}\#, \gamma_2\gamma_2\gamma_1, \beta_3),\end{aligned}$$

for $t_1 \geq 0$, $c_1 > 0$, and $\gamma_2 \neq \lambda$.

Next, we turn to the input suffixes. For all $n \geq 0$, we define the mapping $h_n\colon \{a,b\}^* \to \mathbb{N}$ as follows. If on input $a^n\#w\#a^n$ the pushdown content of M after processing the prefix $a^n\#$ is γ then h_n maps w to the number of symbols at the bottom of the pushdown that are no more seen in the remaining computation on suffix $w\#a^n$. Clearly, if all symbols from γ are touched during the remaining computation then h_n maps to zero.

We now want the show that $h_n(w) < |\gamma_2\gamma_1|$ for all n large enough and all $w \in L_{bin}$. By contradiction, we assume that for some n large enough there exists a $w \in L_{bin}$ such that $h_n(w) \geq |\gamma_2\gamma_1|$. Since the computation

$$\begin{aligned}(\lambda, q_0, a^n\#w\#a^n\lhd, \bot, \lambda) &\vdash^* (a^{t_1+c_1}, q_1, a^{n-t_1-c_1}\#w\#a^n\lhd, \gamma_2\gamma_1, \beta_2)\\ &\vdash^* (a^n\#w\#a^n, q_f, \lhd, \gamma, \beta)\end{aligned}$$

is successful and $h_n(w) \geq |\gamma_2\gamma_1|$, the computation

$$(\lambda, q_0, a^{n+c_1}\#w\#a^n\lhd, \bot, \lambda) \vdash^* (a^{t_1+c_1}, q_1, a^{n-t_1}\#w\#a^n\lhd, \gamma_2\gamma_1, \beta_2)$$
$$\vdash^* (a^{t_1+2c_1}, q_1, a^{n-t_1-c_1}\#w\#a^n\lhd, \gamma_2\gamma_2\gamma_1, \beta_3) \vdash^* (a^{n+c_1}\#w\#a^n, q_f, \lhd, \gamma, \beta')$$

is successful as well, a contradiction.

We conclude that, for all n large enough and all $w \in L_{bin}$, $h_n(w) < |\gamma_2\gamma_1|$.

The next step is to show that there is some constant k such that the value of $h_n(w)$ is reached for the last time, that is, the height of the pushdown is $h_n(w)$ for the last time, when the input head is scanning one of the last k input symbols. By way of contradiction, we distinguish two cases, namely, the input head is scanning some symbol of the factor $w\#$ or the input head is scanning some symbol a of the suffix a^n and the distance to the end of input is larger than k. In the first case, we assume that the input head is scanning some symbol of the factor $w\#$, say in a configuration $(a^n\#u, p_1, v\#a^n\lhd, \gamma, \beta)$, where $w = uv$ and $|\gamma| = h_n(w)$. Note that the computation continues from this configuration successfully and, by definition, u belongs to L_{bin}. We consider the input $a^n\#u\#a^n$ and, thus, the computation continuing in configuration $(a^n\#u, p_1, \#a^n\lhd, \gamma, \beta)$. Assume the configuration after the next move of the input head is $(a^n\#u\#, p_2, a^n\lhd, \gamma', \beta')$. We derive $|\gamma'| \leq h_n(w) + 1$. Since the remaining computation is on unary input a^n, a rough estimation gives that there are at most $|\Gamma|^{|\gamma'|} \cdot |Q|$ different possibilities for the topmost $|\gamma'|$ pushdown symbols and a state. Since n has been chosen large enough, the computation continues in cycles with respect to the state and the topmost $|\gamma'|$ pushdown symbols. The maximal length of the cycle solely depends on $|Q|$ and $|\Gamma|$. So, there are at least two different numbers n_1 and n_2 such that the computations on inputs $a^n\#u\#a^{n_1}$ and $a^n\#u\#a^{n_2}$ are both successful, a contradiction that concludes the first case.

In the second case, we assume that the value of $h_n(w)$ is reached for the last time when the input head is scanning some symbol a of the suffix a^n, say in a configuration $(a^n\#w\#a^{s_1}, p_1, a^{s_2}\lhd, \gamma, \beta)$, where $n = s_1 + s_2$ and $|\gamma| = h_n(w)$. Again, the computation continues from this configuration successfully. Since $h_n(w) < |\gamma_2\gamma_1|$ the value of $h_n(w)$ is bounded from above by a constant k_0 that depends solely on $|Q|$ and $|\Gamma|$. As in the first case the remaining computation is on unary input a^{s_2} and, thus, there are at most $|\Gamma|^{k_0} \cdot |Q|$ different possibilities for the topmost $|\gamma|$ pushdown symbols and a state. This number of possibilities is also a constant, say k, that depends solely on $|Q|$ and $|\Gamma|$. Assume now $s_2 \geq k$. Then the argument of the first case applies that says that the computation continues in cycles with respect to the state and the topmost k_0 pushdown symbols, where the length of the cycle is at most k. So, extending the suffix a^n by as many symbols a as are read during a cycle, yields a successful computation on an input that is not the first component of any pair in 3Occur, a contradiction.

Summarizing so far, we conclude that, for all n large enough and all $w \in L_{bin}$, $h_n(w) < |\gamma_2\gamma_1|$ and the value of $h_n(w)$ is reached for the last time when the input head is scanning one of the last k input symbols, where k is some constant.

The last step in the proof is to apply incompressibility arguments to obtain a contradiction to the assumption that the REV-w-DPDT M realizes 3Occur.

To this end, we choose a (long enough) word $w \in \{0,1\}^*$ with $C(w) \geq |w|$, consider a successful computation of M on $a^{|t(w)|}\#t(w)\#a^{|t(w)|}$, and show that w can be compressed via M.

First, we describe a program P which reconstructs w from a description of M, the length $|t(w)|$, the last two letters z_1z_2 of $t(w)$ as well as the state p_1, the number s_2 of remaining input symbols, and the pushdown content γ of the configuration in which the value of $h_n(w)$ is reached for the last time.

The program P first simulates the remaining computation of M beginning in the configuration in which the value of $h_n(w)$ is reached for the last time. That is, it simulates the computation $(\lambda, p_1, a^{s_2}\lhd, \gamma, \lambda) \vdash^* (a^{s_2}, q_f, \lhd, \gamma', \beta)$, where M halts accepting in the latter configuration. Next, P enumerates all words x of length $|t(w)|$ from L_{bin} whose last two letters are z_1z_2 and simulates M on each candidate $a^{|t(w)|}\#x\#a^{|t(w)|}$. If the simulation halts accepting in state q_f with pushdown content γ' and output $a^{|t(w)|}\#a^{|t(w)|}\#a^{|t(w)|}$ we have $x = t(w)$ and, thus, reconstructed w.

In order to show the correctness of the reconstruction, assume contrarily that there is another input $y \neq t(w)$ in L_{bin} ending with z_1z_2, for which the simulation of M halts accepting in state q_f with pushdown content γ' and output $a^{|t(w)|}\#a^{|t(w)|}\#a^{|t(w)|}$. Let $n = |t(w)|$ and $x = x_1x_2\cdots x_n$ and $y = y_1y_2\cdots y_n$. From the accepting configurations, run M *backwards* (using the reversibility of M) for as long as the suffixes of x and y are identical. This eventually reaches configurations

$$(a^n\#x_1x_2\cdots x_{i-1}, p, x_ix_{i+1}\cdots x_n\#a^n\lhd, \gamma_1, \beta_1)$$

and

$$(a^n\#y_1y_2\cdots y_{i-1}, p, y_iy_{i+1}\cdots y_n\#a^n\lhd, \gamma_1, \beta_1)$$

differing only in their inputs, such that $x_j = y_j$ for $i \leq j \leq n$, and $x_{i-1} \neq y_{i-1}$. Since x and y both end with z_1z_2, we have $i \leq n-1$, and since $x \neq y$, we have $i \geq 2$.

Now, x_i is either a or b, say b. That is, $x_i = y_i = b$ and $x_{i-1} \neq y_{i-1}$. This implies that precisely one of x_{i-1} and y_{i-1} is b, say, $x_{i-1} = b$ and $y_{i-1} = a$.

The word $y_1y_2\cdots y_ib$ belongs to L_{bin}. Moreover, it can be extended to some word $y_1y_2\cdots y_iby'_{i+2}\cdots y'_n$ that belongs to L_{bin} as well. Since M is deterministic we obtain the computation

$$\begin{aligned}(\lambda, q_0, a^n\#y_1y_2\cdots y_iby'_{i+2}\cdots y'_n\#a^n\lhd, \bot, \lambda)& \\ \vdash^* (a^n\#y_1y_2\cdots y_{i-1}, p, y_iby'_{i+2}\cdots y'_n\#a^n\lhd, \gamma_1, \beta_1)&\end{aligned}$$

that must continue successfully. Therefore, the computation

$$\begin{aligned}(\lambda, q_0, a^n\#x_1x_2\cdots x_iby'_{i+2}\cdots y'_n\#a^n\lhd, \bot, \lambda)& \\ \vdash^* (a^n\#x_1x_2\cdots x_{i-1}, p, x_iby'_{i+2}\cdots y'_n\#a^n\lhd, \gamma_1, \beta_1)&\end{aligned}$$

must continue successfully as well. However, since $x_{i-1} = x_i = b$ the word $x_1x_2\cdots x_{i-1}x_iby'_{i+2}\cdots y'_n$ contains the factor bbb and, thus, does not belong

to L_{bin}, a contradiction. We conclude that program P, in fact, reconstructs $t(w)$ and, thus, the word w.

Finally, we determine the Kolmogorov complexity $C(w)$ of w. Recall that P reconstructs w from a description of M, the length $|t(w)|$, the last two letters z_1z_2 of $t(w)$ as well as the state p_1, the number s_2 of remaining input symbols, and the pushdown content γ of the configuration in which the value of $h_n(w)$ is reached for the last time. So, $C(w)$ is given by the length of the binary representation of these ingredients. Let $|P|$ be the constant size of the program P, and $|M|$ be the constant size of M. The length of $t(w)$ can be written with $O(\log(|w|))$ bits and the last two letters z_1z_2 with $O(2 \cdot \log(|\Sigma|))$ bits. For the state p_1, $O(\log(|Q|))$ bits are sufficient. Since $s_2 \in O(|w|)$, it is represented by $O(\log(|w|))$ bits. Finally, recall that the length of the pushdown content γ is bounded from above by the constant k_0. So, $O(k_0 \cdot \log(|\Gamma|))$ bits are sufficient to represent γ. In total, we obtain

$$\begin{aligned} C(w) \in |P| + |M| + O(\log(|w|)) + O(2 \cdot \log(|\Sigma|)) + O(\log(|Q|)) + O(k_0 \cdot \log(|\Gamma|)) \\ = O(\log(|w|)) \subseteq o(|w|). \end{aligned}$$

We conclude $C(w) < |w|$, for w long enough, contradicting that $C(w) \geq |w|$. Therefore, M cannot realize 3Occur. □

The relations between the families of transductions shown are summarized in Figure 5.

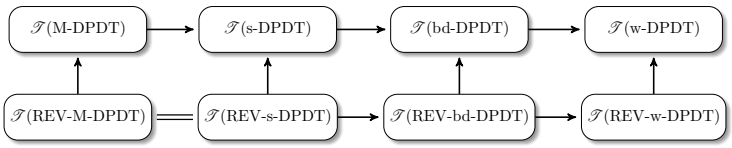

Figure 5: Relations between the families of transductions discussed, where an arrow denotes a proper inclusion. All families which are not linked by a path are pairwise incomparable.

Acknowledgments

This work was supported by the European COST Action IC 1405: Reversible Computation – Extending Horizons of Computing.

References

[1] B. Guillon, M. Kutrib, A. Malcher, L. Prigioniero, Reversible pushdown transducers, in: DLT 2018, Vol. 11088 of LNCS, Springer, 2018, pp. 354–365.

[2] C. H. Bennett, Logical reversibility of computation, IBM J. Res. Dev. 17 (1973) 525–532.

[3] Y. Lecerf, Logique mathématique: Machines de Turing réversible, C. R. Séances Acad. Sci. 257 (1963) 2597–2600.

[4] D. Angluin, Inference of reversible languages, J. ACM 29 (3) (1982) 741–765.

[5] A. Kondacs, J. Watrous, On the power of quantum finite state automata, in: FOCS 1997, IEEE Computer Society, 1997, pp. 66–75.

[6] J.-E. Pin, On reversible automata, in: Latin 1992: Theoretical Informatics, Vol. 583 of LNCS, Springer, 1992, pp. 401–416.

[7] M. Holzer, S. Jakobi, M. Kutrib, Minimal reversible deterministic finite automata, Int. J. Found. Comput. Sci. 29 (2) (2018) 251–270.

[8] G. J. Lavado, G. Pighizzini, L. Prigioniero, Minimal and reduced reversible automata, J. Autom. Lang. Comb. 22 (1-3) (2017) 145–168.

[9] G. J. Lavado, L. Prigioniero, Concise representations of reversible automata, Int. J. Found. Comput. Sci. 30 (6-7) (2019) 1157–1175.

[10] M. Kutrib, A. Malcher, Reversible pushdown automata, J. Comput. System Sci. 78 (2012) 1814–1827.

[11] J. Berstel, Transductions and Context-Free Languages, Teubner, Stuttgart, 1979.

[12] A. V. Aho, J. D. Ullman, The theory of parsing, translation, and compiling, Vol. I: Parsing, Prentice-Hall Inc., Englewood Cliffs, New Jersey, USA, 1972.

[13] M. Kutrib, A. Malcher, C. Mereghetti, B. Palano, Descriptional complexity of iterated uniform finite-state transducers, in: DCFS 2019, Vol. 11612 of LNCS, Springer, 2019, pp. 223–234.

[14] M. Kutrib, A. Malcher, C. Mereghetti, B. Palano, Descriptional complexity of iterated uniform finite-state transducers, Inf. Comput.

[15] S. Bensch, J. Björklund, M. Kutrib, Deterministic stack transducers, Int. J. Found. Comput. Sci. 28 (5) (2017) 583–602.

[16] O. H. Ibarra, H. Yen, On the containment and equivalence problems for two-way transducers, Theor. Comput. Sci. 429 (2012) 155–163.

[17] M. Kutrib, A. Malcher, One-dimensional cellular automaton transducers, Fund. Inform. 126 (2013) 201–224.

[18] L. Dartois, P. Fournier, I. Jecker, N. Lhote, On reversible transducers, in: ICALP 2017, Vol. 80 of LIPIcs, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017, pp. 113:1–113:12.

[19] M. Kutrib, A. Malcher, M. Wendlandt, Transducing reversibly with finite state machines, Theor. Comput. Sci. 787 (2019) 111–126.

[20] H. B. Axelsen, S. Jakobi, M. Kutrib, A. Malcher, A hierarchy of fast reversible Turing machines, in: RC 2015, Vol. 9138 of LNCS, Springer, 2015, pp. 29–44.

[21] M. Kutrib, A. Malcher, M. Wendlandt, Reversible queue automata, Fund. Inform. 148 (2016) 341–368.

[22] Y. Igarashi, A pumping lemma for real-time deterministic context-free languages, Theor. Comput. Sci. 36 (1985) 89–97.

[23] M. Li, P. M. B. Vitányi, An Introduction to Kolmogorov Complexity and Its Applications, Springer, 1993.